DOCUMENT RESUME

ED 249 921 IR 011 340

AUTHOR Sheingold, Karen; And Others
TITLE Chameleon in the Classroom: Developing Roles for Computers. Symposium. Technical Report No. 22.
INSTITUTION Bank Street Coll. of Education, New York, NY. Center for Children and Technology.
PUB DATE Apr 83
NOTE 65p.; For related documents, see IR 011 338, IR 011 343-344, IR 011 352-353, IR 011 359, and IR 011 362. Papers presented at a Symposium at the Annual Meeting of the American Educational Research Association (Montreal, Canada, April 11-15, 1983). This document comprises Technical Report Nos. 12-15 and 20-21.
PUB TYPE Collected Works - Conference Proceedings (021) -- Viewpoints (120) -- Reports - Research/Technical (143)

EDRS PRICE MF01 Plus Postage. PC Not Available from EDRS.
DESCRIPTORS Computer Assisted Instruction; *Computer Software; Design Requirements; *Educational Assessment; *Educational Planning; Educational Research; *Educational Technology; Elementary Secondary Education; Instructional Design; *Microcomputers; *Programing; Word Processing
IDENTIFIERS LOGO Programing Language

ABSTRACT

This symposium includes the following papers: "Software for the Classroom: Issues in the Design of Effective Software Tools" (D. Midian Kurland); "Computers for Composing" (Janet H. Kane); "LOGO Programming and Problem Solving" (Roy D. Pea); "The Computer as Sandcastle" (Jeanne Bamberger); "Learning LOGO Together: The Social Context" (Jan Hawkins); and "Research and Design Issues Concerning the Development of Educational Software for Children" (Cynthia A. Char). The first four papers, preceded by an introduction by Karen Sheingold, are concerned with uses of the computer in the classroom. Kurland argues that tool software may be uniquely important for educators to consider for equipping students to function in this age of information. Kane describes case studies of children using word processors for writing, and comments on the similarities and differences between writing with and without computer technology. Pea discusses several different studies of children learning LOGO programming in classrooms and suggests that it may be important for educators to specify the goals they want to attain by using computers in their classrooms. Bamberger explores the use of programming with students in various musical contexts: rhythm, melody, and composition. The two remaining papers discuss the relationship between the computer and the social context of which it is part. Hawkins discusses peer collaboration as an important context for learning and describes LOGO programming as a facilitator of such collaboration. Char's paper suggests some of the variables that contribute to how software is used, as well as ways of designing software and materials to take account of classroom differences. Two discussants--James A. Levin, of the University of California at San Diego, and Joseph Glick, of the City University of New York--view this collection of papers through their own unique perspectives: Levin looks at the potential of new technology for affecting learning experiences for students and Glick brings a Piagetian developmental perspective to the research. (THC)

CENTER FOR CHILDREN AND TECHNOLOGY

ED249921

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

This document has been reproduced as received from the person or organization originating it.

Minor changes have been made to improve reproduction quality.

Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

"PERMISSION TO REPRODUCE THIS MATERIAL IN MICROFICHE ONLY HAS BEEN GRANTED BY

Denis Newman

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

1983 AERA Annual Meeting
April 11-15, Montreal, Canada

Symposium:
Chameleon in the Classroom: Developing Roles for Computers

Technical Report No. 22

IR011 340

Bank Street College of Education

610 West 112th Street NY, NY 10025

1983 AERA Annual Meeting
April 11-15, Montreal, Canada

Symposium:
Chameleon in the Classroom: Developing Roles for Computers

Technical Report No. 22

CENTER FOR CHILDREN AND TECHNOLOGY
Bank Street College of Education
610 West 112th Street
New York, NY 10025

IRO 113340

CHAMELEON IN THE CLASSROOM: DEVELOPING ROLES FOR COMPUTERS

Technical Report No. 22

(Includes Technical Reports 12-15, 20-21)

Karen Sheingold	1	INTRODUCTION
D. Midian Kurland	4	SOFTWARE FOR THE CLASSROOM: ISSUES IN THE DESIGN OF EFFECTIVE SOFTWARE TOOLS
Janet H. Kane	14	COMPUTERS FOR COMPOSING
Roy D. Pea	25	LOGO PROGRAMMING AND PROBLEM SOLVING
Jeanne Bamberger	34	THE COMPUTER AS SANDCASTLE
Jan Hawkins	40	LEARNING LOGO TOGETHER: THE SOCIAL CONTEXT
Cynthia A. Char	50	RESEARCH AND DESIGN ISSUES CONCERNING THE DEVELOPMENT OF EDUCATIONAL SOFTWARE FOR CHILDREN

Discussants:

James A. Levin	55	
Joseph Glick	57	

INTRODUCTION

Karen Sheingold

Center for Children and Technology
Bank Street College of Education

This symposium has been created by the staff and friends of the Center for Children and Technology of the Bank Street College of Education. At the Center, we conduct research about technology and the many ways in which it may affect processes of learning and development, as well as the life of the classroom. We are interested in better understanding what children bring to and get out of various kinds of computer-related activities, as well as in how to shape such activities for the benefit of students and teachers.

In the title of our symposium, we refer to the computer as chameleon. We mean this in two senses. First, the microcomputer is like a chameleon because there are many uses to which it can be put in the classroom. It is these uses--the computer as programming environment, the computer as writing tool, the computer as provider of drill and practice--and not "the computer" in some global sense which influence what kinds of research questions we can ask, and what kinds of educational hopes and expectations we have about the technology.

One of the ways in which computers can be used in classrooms, although not yet a common use, is as a tool to help students and teachers accomplish some tasks they already want to do--such as writing. Tool software is thought to have practical value; that is, it helps students and teachers do things with greater ease and/or efficiency. It may also offer the possibility of expanding, amplifying, and making qualitatively different the processes by which such tasks are executed. Midian Kurland argues that tool software may be uniquely important for educators to consider for equipping students to function in this age of information. He outlines several uses of the computer, and explores the potential of tool software for the classroom.

The word processor is a particular tool which has been studied--and developed--at Bank Street. Janet Kane describes case studies of children using word processors for writing, and comments on the

similarities and differences between writing with and without computer technology. She discusses the implications of these studies for the classroom use of word processors.

A second way in which computers are used, and currently one of the most widespread uses, is for programming. The Logo language, in particular, has generated excitement in the educational community because of its promise for helping children learn much more than programming. Roy Pea discusses several different studies on children learning Logo in classrooms. His paper raises deep questions about what may be attainable via programming in elementary classrooms. One clear implication is that it may be important for educators to specify just how they wish to use programming in their classrooms--the goals they want to attain--and structure the programming experience with those goals in mind.

Jeanne Bamberger, from the Massachusetts Institute of Technology, believes that programming may be very useful for helping us to know in a more formal way what we already know at an intuitive level. In the domain of music, she has explored this use of programming with students in various contexts. As a teacher, researcher, and musician, she has designed programming experiences which help students understand in new ways what they already know about rhythm, melody, and composition.

The computer uses explored in the first four papers illustrate the first sense in which the computer is a chameleon--its uses take many forms. There is a second sense in which this analogy is important: in the relation between the computer and the social context of which it is a part. The computer is both transformed by and transforms the social life of the classroom. Jan Hawkins has been particularly interested in peer collaboration as an important context for learning, and in Logo programming as a facilitator of such collaboration. Her paper points out some important ways in which using computers for Logo programming has affected the classrooms she is studying. Yet, her speculative note about the ill-defined nature of computing in the work of the classroom is an appropriate caution to those who wish to draw longterm implications from these results.

The classroom social context can have an important effect on how computer software is used. The very same computer language, tool software, or simulation can look remarkably different in different classrooms. Cynthia Char has been conducting research for an innovative project in mathematics and science education at Bank Street, and has had the opportunity to study the same pieces of software used in different classrooms. Her paper suggests some of

the variables which contribute to how software is used, as well as ways of designing software and materials to take account of classroom differences.

The discussants view this collection of papers through different lenses. James A. Levin, of the University of California at San Diego, has also been looking at the potential of new technology for affecting learning experiences for students. He comments on his reading of these effects and their likely longevity. Joseph Glick, of the City University of New York, in contrast, brings a Piagetian developmental perspective to the research. He uses the notion of assimilation to interpret the results, and argues that people's goals and intentions must be taken into account in any adequate formulation of what students will learn when they use computers.

This collection of papers and comments represents, in my view, an important set of insights and issues for educators to consider when planning for the use of technology in schools. In addition, it maps out, refines, and redefines some of the most challenging research questions for those interested in better understanding the potential and limitations of new technologies for influencing learning and education.

SOFTWARE FOR THE CLASSROOM: ISSUES IN THE DESIGN OF EFFECTIVE SOFTWARE TOOLS*

D. Midian Kurland

Center for Children and Technology
Bank Street College of Education

According to current estimates, by the end of 1983 more than 1,000,000 microcomputers will be in place in classrooms across the country. Spurred by lowering costs as well as by pressure from parents and educational policy makers, schools are purchasing computer hardware at an accelerating rate. As the price of peripheral devices such as disk drives and printers also falls, it is becoming possible to envision a time in which every classroom will have its own computer system. While the availability of inexpensive and reliable computer hardware is fast becoming a reality, the future of microcomputer technology in classrooms hinges on appropriately designed application software, and on how teachers are able to incorporate it into their classrooms. How well or how poorly available software helps teachers to accomplish their classroom goals will determine whether computers will be the powerful instructional force which its proponents foresee, or merely another educational innovation that becomes nullified through partial assimilation by the educational system.

How well, then, is educational software keeping pace with the rapid increase in available hardware? Is there software appropriate for the many uses to which a computer can be put that is geared to the user environments of schools? In general, the answer to both these questions is no. If one takes a look at any of the educational computing magazines or talks to teachers struggling with their first computer, a general impression emerges that educational software is often poorly conceived, "buggy," difficult to use, difficult to integrate with the rest of the curriculum, and designed without sufficient regard for the range of needs and abilities of students. While it would be unfair to the authors of the few interesting and well-executed educational programs to brand all educational software as inadequate, the fact

---

*Technical Report No. 15.

remains that the vast majority of software produced for schools is amateurish, unimaginative, or both.

What should a teacher or administrator do about software when the first computers roll through the door? How can nonspecialists make sense of the mountain of available programs? How can they select programs of high quality which fit with the educational philosophy and instructional orientation of their schools? How can software help those teachers who are new to computers better understand their potential applications? Clearly, before teachers and administrators can make intelligent decisions about what software to purchase for their computers, they must have clearer ways of thinking about what a computer is for, and how different types of software reflect these different purposes.

There are at present three ways in which computers are being used in educational contexts--as conveyers of content, as programming environments, and as problem-orientated tools. Implicit in each use is a different model of what a computer is and what its role in the classroom should be. I will briefly discuss the first two models--computer as tutor and computer as programming environment; then I will discuss in more detail the third model--computer as tool. Though currently the most neglected of the three approaches, I will argue that it is this type of use which holds the most promise for computer use in the classroom.

Computer as Tutor

One of the earliest uses of computers in the classroom, and still the most widespread, is as a delivery system for programmed instruction and drill-and-practice activities. The use of computers to deliver "courseware" has been promoted by many of the large educational publishers who want software that supports and looks very much like the textbooks they produce. Such uses of the computer are also promoted by most educational computing magazines and conferences through what they choose to focus their reviews and critiques on. When looking for classroom software, teachers frequently first turn to the published software reviews that are provided by EPIE, Classroom Computer News, Electronic Learning, and other computer-oriented periodicals. These reviews focus almost exclusively on "instructional software" in the form of drill-and-practice programs and/or games. Putting aside the question of how to evaluate software--how to decide which features of different types of programs to evaluate; how to weight pluses and minuses of individual features in arriving at an overall rating; and how to maintain the consistency, quality, and reliability of the reviewers--by following this path to acquiring soft-

ware, the school inevitably ends up with a large number of individual, unrelated programs, each aimed at some specific skill or concept. Since approximately 90% of the programs reviewed for the educational market are based on instructional drill-and-practice formats, these review efforts tacitly support using the computer as the transmitter of predefined content, and placing the student in the role of receiver or respondent.

Computer as Programming Environment

Using the computer to run packaged courseware, regardless of its quality, has many limitations. While it does give the classroom computer something to do, in many cases it is not clear that using the computer in this way has any real advantage over older instructional technologies such as flashcards and workbooks. In contrast, there is a growing contingent of educators who choose to bypass the issue of what software to use by having their students focus all their efforts on learning a programming language such as BASIC, Logo, or PASCAL. This approach has the advantage of being inexpensive, since most computers come with one or more languages installed, and students of all ages can, in theory, be taught the same language using the same software. Further, knowing how to program a computer is a valued skill outside of the school, whereas knowing how to sit through a drill-and-practice lesson is not. Finally, learning to program may teach the student something about logic, reasoning, and problem solving that can potentially be applied in domains far removed from programming.

Using the computer to teach about computers and programming does have its merits. However, this approach provides students and teachers with a rather narrow view of what a computer is for and what it can do. Further, programming, especially for novices, is a difficult activity to put into a context that makes clear the goals and purpose of learning to program. Programming a computer is difficult work. It takes a great deal of time and effort before a student is capable of using his or her programming skills in the service of real problems in other curriculum areas. Commercial software programs exist precisely so that one does not have to spend hours or days writing a program each time there is a problem to be solved. It is unrealistic to deny students the use of commercial software and, at the same time, expect them to use the computer to assist them in their coursework. Thus, unless the teacher has the time and is highly skilled at preparing programming "microworlds," programming tends to remain a separate subject that must be taught in addition to the rest of the curriculum. This puts a significant strain on already

overburdened teachers who may not even have enough time for teaching the traditional subjects such as science and social studies.

Opting for the programming route also runs into the problem that students are introduced to programming in a manner which bears little resemblance to the way programs are actually written by professional programmers. No experienced programmer would ordinarily write a program without a software library of programming utilities and aids. A programming language is merely one part of a complete programming environment that should include program editors, debugging utilities, simulators, graphics and music editing systems, and special input and documenting routines to speed up coding. Using the computer to teach programming does not eliminate the need for software; it simply changes the type of software needed.

Computer as Tool

An alternative way of conceptualizing the role of the computer in the classroom is to think of it as a flexible, reconfigurable tool. A tool, by definition, is anything that a person can use to get a job done faster, better, and/or with less effort. Schools are full of specialized tools to help the student through the curriculum. Math tools such as the protractor, ruler, and compass help the student to acquire the concepts of geometry and measurement. Similarly, pencils, erasers, scissors, and tape are tools that assist the student to produce and revise texts. Just as these traditional instructional tools can be used in the service of a diversity of curriculum goals, the computer equipped with tool software can serve many purposes. The class of tool software consists of programs which turn the computer into a word processor, calculator, music system, data organizer, graphing system, note taker, or bulletin board. None of these types of programs teaches a specific content, but neither are they to be used purely as an adjunct to the regular curriculum. By serving established needs, such as creating and revising a composition or producing a graph of a simple function, tool software is easily understood and assimilated by teachers and students regardless of their prior level of experience.

Rather than replacing the teacher--the implicit goal of the computer-as-tutor model--the computer-as-tool model puts the teacher back into the thick of the educational process, armed with an enlarged array of powerful instructional tools. However, while the idea of using the computer as a tool for many different purposes is appealing on a number of levels, the quality of available software tools poses a serious problem for the educator. Put bluntly, there are almost no software tools which have been produced for the educational market,

and what there are tend to be poorly thought out and difficult to use. However, many extremely powerful and well-executed software tools have been created for the business world. While educational publishers are churning out programs that look like workbook pages, business programmers are producing powerful, yet easy-to-use tools. These software tools not only facilitate traditional business activities, but make possible whole new types of activities that, previously, were either too time consuming and/or too difficult to be practical.

Consider, for example, the VisiCalc program. Throughout the sixties and early seventies, computerized business systems replicated functions done by older technologies (payrolls, inventories, mailing lists, etc.). Similarly, most current educational software replicates the type of drill-and-practice activities previously done with flashcards or worksheets. The early users of business systems had little experience with the capabilities of the technology; hence, they demanded nothing more innovative.

However, when VisiCalc was introduced in the late seventies, it began to change the way the business world thought about the computer. Designed as a general-purpose accounting program, VisiCalc was easy to use, even by those with no previous computer experience. More importantly, it facilitated the manipulation of information in ways that were novel to users. They could now model a wide variety of situations by systematically varying the design of the parameters. The program automatically calculated and displayed the effect of changing one parameter on the remaining variables in the model. This piece of software provided new means for thinking about and manipulating information; simultaneously, it made users aware of the novel functions of the computer.

VisiCalc has been followed by a whole generation of innovative office software. Relational data-base management systems, sophisticated word processors, and electronic mail and other communication systems are examples of programs which, over the past few years, have radically altered information use and management in the business world.

The VisiCalc example yields an important lesson about the interaction of software and context. The software was initially recognizable and usable as something that fit into the work needs of managers and executives. Yet, the experience of exploring and exploiting the potential of the software yielded something much more powerful than numbers arranged neatly on the page--namely, new ways of thinking about business information.

Such developments should also be possible in the educational realm. However, until producers of educational software begin to produce more software tools, it will be necessary for educators to seriously consider products developed for other markets and types of users. For the present, an interesting problem facing educators is how to adapt the professionally developed products created for the business market to the needs of teachers.

What is needed is software which teachers can immediately see as helping to do the current work of the classroom, but which also supports new ways of working, learning, and teaching. An important goal for software being produced for today's classroom is that it teach the child, but also illustrate the open-ended potential of this technology for teachers.

A word processor is a wonderful example of just such a program. It does not introduce a new activity into the classroom. Writing is simply accomplished through a different medium. Studies under way in our laboratory indicate that with the right software, students can start writing on the computer the day it comes into the room. Stories that were started on paper can be moved onto the word processor with little trouble. The teacher need not invent new activities, new projects, or take time to teach a new subject before students can use the computer.

Compared to the many other potential uses of the classroom computer, word processing is relatively easy for noncomputer-experienced teachers to try. However, while using a word processor may initially seem like simply substituting the keyboard for the pen, once students start using the word processor, unanticipated effects begin to appear. For example, in a recent study, a teacher reported that when the computers came into her classroom, she found herself immediately starting to correct the students' papers in ink instead of pencil. Asking them to revise their work no longer required a major production effort from the students. Marking up a text did not mean requiring the student to recopy it. This immediately freed the teacher to request more specific revisions from the students. Students reported, and their writing results appeared to indicate, that they wrote longer pieces and revised their writing more often when they could use the word processor. There also was a greater tendency to collaborate on a piece of writing and to read each other's writing since to do so was now much easier with the availability of typed copy and text displayed on a large screen.

The teacher discovered that conferences at the computer with the student's text on the screen was a wonderful way to display, discuss,

and revise a piece of writing. Changes could be made as they were discussed. At the end of the conference, the student had a typed, revised copy instead of a marked-up piece of paper to be recopied. Thus, working with a word processor began to change how the writing process took place in the classroom on a number of levels, in addition to facilitating the mechanical aspects of producing a text.

Besides creating new ways for teachers and students to think about certain activities, software tools can also be used in more traditional ways when appropriate. Use of a word processor need not be limited simply to composition. By preparing paragraphs, sentences, or word lists and then storing them in files on a disk, a teacher can create grammar, spelling, and style exercises tailored to the needs of individual students or to the whole class. This type of tool software can accomplish more effectively what a stack of drill-and-practice language arts programs are required to do, while at the same time acquainting the student with a tool that can be applied to a wide range of problems in the future.

Word processors are good examples of software tools which, though originally developed for the office environment, can be and have been brought into the classroom and utilized effectively. However, software designed for one environment is not always well suited for another. In the case of word processors, many have been designed around the needs of secretaries. These programs are intended to be used on a daily basis to enter, revise, and print someone else's text using powerful but complex editing and formatting commands. Thus, these word processors are often unnecessarily complicated--difficult to learn or to remember how to use. Similarly, systems for office use may include functions that, while useful in the office, may not be educationally sound in the classroom. Consider the example of programs which can catch spelling errors in documents.

The current state of the art allows the user of a microcomputer to check a document against a 20- to 50-thousand word dictionary stored on a floppy disk. The program can mark each word in the document that is not in its dictionary, suggest a more likely spelling, and make the correction with or without the student's assistance. In addition, such programs can evaluate grammar, word usage, sentence structure, tone, and a number of other stylistic parameters which might be of interest to the writer. Because currently available computer software can do such tasks for the writer, educators are forced to think hard about what it is we want tool software to do. Will the use of these spelling checkers seriously lower the spelling ability of students? If it does, does this really matter any more since we have spelling checkers? Or should schools insist on spelling checkers

which simply report how many suspicious words are in a text or in a paragraph and leave it to the student to try to find them? Or should the dictionary be available only for students to look up specific words that they are unsure of as they write?

In light of the almost total lack of research into such issues, it seems that the best option at this point is for software designers to implement each of these levels of spelling checking, and let the individual teacher set the program for the type of assistance the students can have. In this way, the teacher could make available an on-line dictionary while the student is writing, and let the student use the automated proofreading facilities only after final drafts of a piece of writing have been accepted.

Today, if a teacher uses a spelling program in the classroom, he or she has little control over the form of assistance it provides. Nonetheless, until spelling checkers are developed that are sensitive to instructional needs, the currently available spelling checkers, if used creatively, can be effective in helping students with their writing.

Word processors and spelling checkers are examples of tools which primarily facilitate the production end of the writing process. Different types of existing software tools can be used to facilitate other aspects of writing. Data-base management systems, for example, can be employed effectively as prewriting tools to help the student collect, organize, and discover patterns and relationships in information before beginning a report. Simple to learn and use, data-base systems have existed for the popular microcomputer for the past several years. They can be used collectively by a class to enter information on some topic--for example, marine animals--after which students, individually or in small groups, can interrogate the data base as a means of exploring such questions as: What mammals live in the ocean? Do carnivores tend to live close to the shore? What kinds of sea animals have commercial value? and so on. Reports generated by data-base systems can then be used as notes for the students when they write up their findings.

Tools such as data-base management systems do not teach about a subject, nor do they directly lead the student into entering the best information in the best way or in framing the best searches for that information. They simply make it more possible for that to happen. However, not all writing tools need to be this open-ended. For example, the Planner, one of the Quill Writing Programs being developed at Bolt Beranek and Newman, Inc., helps students to consider the types of information to include when doing a specific type of writing. Planner facilitates the precomposing stage of the writing

process by helping the student decide what content to include in his or her writing through a process of electronic brainstorming. The program is designed with a series of templates which can be modified or added to by the teacher or students. Each template is used for a different type of writing or subject area, such as a movie review, persuasive essay, or newspaper story. The student selects one of the Planner templates, and then is asked a series of pertinent questions about the selected topic. For example, the teacher can set up a Planner template for marine animal reports. When the student uses the Planner program, a series of questions appears on the screen which might request that the student type in the topic, the animal to be discussed, two of its characterizing features, two ways in which it is of importance to man, what it likes to eat, where it lives, and what is special about it. When the student is finished with the Planner, a copy of the prompts and the student's answers is printed out as an outline to be used when writing the report.

Conclusion

While the focus of this paper has been on tools to facilitate the writing process, powerful software tools that put the user in control of the computer exist for many purposes. They are available for music, graphics, math, communications, and many other domains. Unlike most of the software that is finding its way into schools, tool software represents an open-ended potential limited only by the imagination of the teachers and students who use it. A program such as Spelling Demons, which drills students on the 100 most frequently misspelled words, can never do any more than that. However, a word processor for approximately the same price can be used not only for spelling lessons, but for other activities as well by children of most grade levels and abilities.

The greatest strength of the computer lies in the fact that it can be quickly reconfigured to be many different kinds of tools--word processor, graphics editor, data collector, music composition aid, calculator, and proofreader, as well as tutor or tester. When considering what software to use with the classroom computer, the issue is not simply what content area the software addresses, but how the content is approached, how much control the student has, and how wide a range of uses the program serves. In order to improve and better understand the use of software in classrooms, research is needed to analyze the current software tools being used in schools; to evaluate the many other software tools which are currently available, irrespective of their intended audience; and to develop better means of integrating software tools into the classroom context. As our society moves into the information age in which access to information will be

mediated more and more through computers, teaching students to use software tools effectively will become an increasingly important role for schools. It is time for both educators and software producers to recognize this, and to begin a serious effort to develop the tools that will provide access to this rapidly approaching world.

COMPUTERS FOR COMPOSING*,**

Janet H. Kane

Center for Children and Technology
Bank Street College of Education***

Introduction

Experienced writers compose in many drafts. They revise early drafts structurally and conceptually, and modify individual words and sentences in later drafts (Bartlett, 1981; Calkins, 1979; Graves, 1978; Nold, 1981). The goals of this process are to explore and clarify ideas and to create a written text that communicates effectively with the intended audience. For expert writers, the composing process, with its components of planning, writing, and revising, is reader-directed and iterative, incorporating a wide variety of strategies for revision.

Students' models of the composing process contrast sharply with those of skilled writers. Students strive to make their compositions "right" the first time (Shaughnessy, 1977). Before starting to write, they mentally organize their ideas. Their goal is to tell what they know rather than to refine their understanding or to have a particular effect on the reader (Flower, 1979; Scardamalia & Bereiter, 1982). When students write, they start with the first sentence and continue linearly until they are finished. Except for corrections in spelling and punctuation, they rarely modify their texts (Bartlett, 1981; Emig, 1971). Most only do two drafts, the second merely a neater and more legible copy of the first. For student writers, the composing process, with its stages of planning, writing, and revising, is sequential, with revisions primarily limited to corrections in spelling and punctuation and changes in wording or phrasing.

---

*Technical Report No. 21.

**The research reported in this paper was funded by the Richard Lounsbery Foundation.

***Dr. Kane can now be reached at 2661 Princeton Road, Iowa City, Iowa 52240.

Once a draft is written, changes are mechanically difficult. Erasing makes the paper look sloppy. Adding words, sentences, or paragraphs, or reordering the text is impossible without tedious recopying or cutting and pasting. Once a text is written, it may as well be carved in stone.

While teachers describe the multiple drafts of expert writers, they seldom require students to do several drafts of the same paper. Usually, the teacher will critique one assignment, and assume that the students will be able to incorporate the suggestions when writing on another topic.

What if text were easy to change? Would students compose more like expert writers if the mechanical difficulties of modifying their written texts were minimized? Would teachers expect students actually to make revisions of their papers?

Word processing technology was developed to facilitate the production and revision of printed material. With it, words, phrases, sentences, and even paragraphs can easily be inserted or deleted from written text by simply pressing a few keys.

While the cost of dedicated word processing equipment puts it out of reach of most schools, relatively inexpensive software is available for use with microcomputers. Microcomputers are already in many schools across the nation, though they are usually used for mathematics or programming. With word processing software, the microcomputer can also be a tool for students in the humanities.

In the fall of 1981, under a grant from the Richard Lounsbery Foundation, we began to explore the potential of word processing technology for promoting the development of composing skills, with special emphasis on revision.

Software Selection

Our first task was to find word processing software that seemed suitable for eighth graders. We reviewed many of the commercially available systems and were impressed by the range of capabilities. However, we felt that most of them were not appropriate for writers with little or no experience with computers and limited opportunities to learn about them. Most systems were more applicable to businesses than to authors.

In most offices, one person composes a text and a different person types it; still others may revise and edit it. A word processing

system designed to format and change texts that others have written is necessarily quite different from one designed to be used while composing.

The most obvious difference is in the cognitive demands placed on the user. Typically, the individual who is putting the text into presentable form is only minimally concerned with its meaning. This staff member does not struggle with the problem of how effectively to reach the intended audience. Word processing specialists, in the day-to-day execution of their work, become familiar with the intricacies of a system and use the commands automatically. They can interrupt their typing to consult operator reference guides for the individual steps of a complex procedure. In contrast, writers are primarily occupied with composing coherent and compelling texts. If writers must attend to the complexities of using a word processing system, the quality of the writing may be compromised.

We did find a few word processing systems that seemed relatively easy to use. None, however, seemed ideal for student writers. Each had advantages as well as what seemed to us to be serious drawbacks. Three of these systems were reviewed by a panel of students. Twelve eighth graders learned the fundamentals of each system, and used it to revise a short passage and to compose a short text. After using all three systems, the students evaluated them and described the features they preferred as well as those that made the system difficult to use. On the basis of their comments, APPLEWRITER was selected for the research.

Research Design

This exploratory study was designed to provide insight into two questions:

A. Can students use this word processing technology? Is it sufficiently easy to use so that it will not interfere with composing?

B. Will students become more fluent and more flexible writers by using word processing technology?

The Participants

One teacher and five eighth graders (one girl and four boys) from the Bank Street School for Children participated in this exploratory study. (Another girl was selected, but missed most of the classes because she had chicken pox.) The teacher was an experienced teacher of writing, as well as a writer herself. She valued students'

critiquing and revising their own and others' work, and was interested in learning to use word processing software. The teacher had not had any previous experience with computers.

The students were selected from a larger group of volunteers. The participants represented a wide range of writing skill. Some wrote very fluently; others continually struggled with expressing ideas and emotions through written language.

Two of the students had used computers before. The girl had a computer in her home and her father had taught her to write simple programs in BASIC. One of the boys occasionally did some simple programming on a computer at his father's office. The other three students' experience with computers was limited to video games.

In addition to the teacher and students, two researchers attended each class session to help the teacher and students use the equipment, as well as to study and record how it was used.

The Setting

Students used the word processing system as part of a 5-week minicourse that met for two 45-minute sessions each week. They came to a computer workroom where each had a computer, a video monitor, and a disk drive. Two printers were also available. At each session, students who were first to arrive had their choice of equipment.

At the first session, the teacher briefly explained the purpose of the class. The researchers demonstrated the use of the word processor, and each student was given a disk and a reference sheet describing commonly used commands. Then students were encouraged to start composing. They had their choice of composing at the computer or writing on a sheet of paper and then entering the text.

As students wrote, the teacher moved around the room, observing their writing and asking or answering questions. The two researchers were also available to demonstrate how to use a particular procedure or to help with equipment problems.

Ten minutes before the end of each class, students were encouraged to finish their work for the day and to print out a copy if they wanted one. Sometimes the teacher asked a student to make a copy for her to look over and comment on for the next class.

The Writing Curriculum

At the Bank Street School for Children, writing is encouraged and valued as a means of exploring and expressing ideas and feelings. Students receive honest and constructive criticism of their work from both peers and teachers, and learn about features of effective text as well as strategies for revision. Students write frequently and regularly, making daily entries in their personal journals and preparing 3- to 4-page reports for each curricular unit.

In the minicourse, students were encouraged to write expressively: they were to write whatever came to mind, and to continue without concern for structure or transition. Their goal was to explore topics through writing, and then to develop a plan for what they wanted to communicate.

Data Collection

Interviews. Prior to the minicourse, all students were individually interviewed about their models for composing, their development as writers, their standards for effective writing, and their strategies for revision. After completing the minicourse, students were again interviewed to learn about their reactions to word processing technology as a medium for composing.

Observations. At each class session, students' activities were recorded at 5-minute intervals. In addition, one or two students were observed as they composed and/or revised their written text, and the kinds of changes they made were recorded.

Writing samples. Copies of each student's files were made at the end of every class session, and the files were compared across sessions.

Composing with the Computer

In the minicourse, students generally spent five to six class sessions composing a piece (or, in the case of one student, deciding to abandon it). The first one or two sessions usually involved very little writing; students seemed to be mentally exploring potential topics. Once they had found a topic, the next two to four sessions were spent writing the body of the paper. Students seldom made typographical errors because they typed very slowly and deliberately, but when they did, they usually corrected them immediately, using the function key for deleting.

The girl in the minicourse was a fluent writer. She composed a lengthy narrative about a vacation adventure, approaching her writing enthusiastically and with a clear idea of what she wanted to say. She was concerned with making the story interesting and entertaining, and made several changes to make the text more appealing.

She made quite a few revisions in her text, spending part of each session modifying what she had written previously. She added and deleted sentences within her text, inserted words to clarify meanings, and made substitutions of words and phrases. Once she brought in a printout that she had revised at home and implemented most of the changes she had planned. In another session, she changed her text in ways prompted by the teacher's comments.

The other minicourse students seemed to compose through association. After writing a sentence, they would read it over and decide whether to keep or delete it. Then they would sit for a long time contemplating their next sentence. They did not seem to have a general goal with which to guide their production.

From one session to the next, these students would simply extend their texts. The only changes they made to previously written texts would be to correct spelling or punctuation. After several sessions, two students seemed to discover their meaning and organized their text into paragraphs. A third student didn't finish, but simply stopped after urging from the teacher. A fourth was never satisfied with his text, and spent the entire course starting over rather than developing his composition.

All students started one or two other texts, but none had time to finish them during the minicourse. The composing process seemed similar to that observed with the first text.

Conclusions

A. Can students use this word processing system?

All students were able to use the technology for composing, and all said they would like to use it regularly for both school assignments and personal writing. Students preferred the computer to pencil and paper. Throughout the course, pencils and paper were available on a table in the center of the room. Only once did a student choose to move away from the computer and work with a pencil.

Generally, students composed with the computer as they composed with pencil and paper. The new technology was assimilated to their

models for composing. Production was primarily linear and sequential. Most revisions were corrections in punctuation or spelling, though one student added, deleted, and reordered sentences within her text.

None of the students was a proficient typist, but two had had some instruction in touch-typing and knew the positions of the fingers on the keyboard. The other students typed very slowly and laboriously, searching for each key and using only one finger. Even these students, however, were persistent about using the technology for composing. Students were asked about their typing skills in the post-course interview. While most said they would like to learn to type better, none felt that the lack of typing skill was an impediment to composing with the computer.

B. Will students become more fluent and more flexible as writers by using word processing technology?

The 10-session minicourse was too brief to effect any fundamental changes in students' models for composing. These models had been developed over many years of schooling and could hardly be significantly modified in the equivalent of a mere six hours of instruction. However, the results of the minicourse point to six ways in which the technology may be used to support students' development as writers.

1. Students spend more time composing when they use a word processing system. Students' skills may improve simply as a result of spending more time writing. Using the computer, students created, extended, and/or revised their texts across five or six sessions. This contrasts dramatically with their reports that, with pencil and paper, they usually composed a paper in a single sitting of 30 to 45 minutes. The students were also intensively involved with composing at the computer. The teacher had planned for them to read their work to each other, but she found that she could not get them to stop working. At many of the sessions, the room was eerily quiet; students were so absorbed with their own composing that they didn't talk to each other.

2. Students feel free to explore their ideas in writing because deleting is easy, even from the middle of the text. All students in the minicourse used the word processor to delete sentences and paragraphs. In part, this was the result of the teacher's encouragement to write rather than to delay until the paper was entirely thought out. However, the fact that students were able to suspend judgment before writing demonstrates that they did feel freed from the overwhelming constraints imposed by trying to make the written text

perfect the first time. Usually, students did not delete sections of text until they had written most or all of it. Knowing that the text was easy to change allowed the students to ignore some of what they wrote and to concentrate on other sections. They seemed to prefer the more transient CRT display to the permanent printout. After the first session, students seldom made a copy until they were finished with the text.

3. When using the computer, students consider the overall structure of their text. In precourse interviews, students said they never reorganized a handwritten text unless they were told to by their teacher. In the minicourse, all four students who finished a piece modified the paragraph structure. Two students initially had no paragraphs in their compositions, and spent some time dividing their work into paragraphs. Two others improved their work by rearranging the order of their paragraphs.

4. Using word processing technology may facilitate use of the revision strategies students have already learned, eventually resulting in their automatic application. When revising their texts, students most often choose to modify individual words or phrases, often adding to or changing them in order to describe a character, a setting, or an event more vividly or more particularly. These changes did not happen very often--perhaps ten words in an entire composition--but they were effective.

5. Students will be motivated to learn new strategies for evaluating and revising their texts because changes are easy to execute with a word processor. Judging from their interview comments, students had few explicit strategies for revision. Their main goal was to write what they were thinking or feeling. When asked, "How do you know when you've finished writing?" no student could respond. Most did remark, however, that producing an ending or conclusion was one of the most difficult parts of composing. Students also reported that they did not usually think about their audience as they wrote. One student said that thinking about the audience made it more difficult to write, so he preferred to concentrate on his message. Text features such as comprehensibility, enticingness, persuasiveness, and memorability were not considered by most of these eighth-grade writers. (See Bruce et al., 1982 for a discussion of these criteria for communicative effectiveness.)

6. The computer can influence the extent of collaboration while writing. Initially, we hypothesized that with a computer, composing would become more public. Displaying the text on the screen would invite others to read it, and would set the stage for peer discussions

of a text. We felt that, through such critiquing, students would *develop new standards for evaluating texts and acquire new procc-* dures for revision. Contrary to these expectations, writing became less public in the minicourse. Students were so involved with their own texts that they seldom spoke with others. Only once did a student ask another to read his work.

On reflection, we feel that this occurred because students had their own computers and wanted to use this resource as much as possible. If a small number of computers are in a classroom, it will be easier to arrange for two or more students to use them collaboratively. However, if the computer is located in a resource room and students are sent individually to work with the equipment, the computer may undermine attempts to promote collaboration while composing.

Discussion

Word processing systems vary enormously in their complexity and ease of use. After reviewing a wide range of software, we concluded that many systems required too much time to learn to be practical for students and teachers with little or no previous experience with computers. In this research, we studied students' use of a system that was selected precisely because a review panel of eighth graders judged it as easy to use.

Students who used the selected word processing system for ten class sessions did find it easy to use. A major conclusion from this exploratory study is that, initially, students assimilated the technology and applied it to their model for composing. They used it as they used pencil and paper. Production was primarily linear and sequential. Most revisions were corrections in spelling and punctuation, although occasionally a single word or phrase was inserted or replaced.

Students also capitalized on some of the features of the technology. It seems that their models of composing might be restructured to accommodate to the technology. For example, all students modified the organization of their texts, even though they rarely reorganized handwritten texts.

The word processor cannot teach students to be better writers; it only provides a means to effect changes more easily. It will not respond to the author's message. It cannot critique, nor can it recommend particular changes, or judge whether a change is an improvement (but see the Writer's Workbench for a prototype of such

a system--Macdonald, Frase, Gingrich & Keenan, 1982). Unless students have standards of good writing and can evaluate and revise their own work in terms of these standards, changes will not be improvements.

As Donald Graves' work documents (Graves, 1983), students develop as writers when their teachers value students' expressing their own ideas, discuss students' writing with them, and instruct them in the effective use of written language. In successful writing programs, attention is on the meaning the author intends to convey and the meaning the reader will construct from the text.

Although writing skills develop as students communicate through writing, the word processor may prove to be a useful curricular tool. With a tool that eliminates the tedious recopying that is now part of revising, students may be eager to develop strategies for evaluating and improving their texts.

References

Bartlett, E. J. Learning to write: Some cognitive and linguistic components. A report to the Ford Foundation, January 1981.

Bruce, B., Collins, A., Rubin, A. D., & Gentner, D. Three perspectives on writing. Educational Psychologist, 1982, 17, 131-145.

Calkins, L. M. Case study of a nine-year-old writer. Durham, NH: University of New Hampshire Writing Process Lab, 1979.

Emig, J. The composing process of twelfth graders (Research Report No. 13). Champaign, IL: National Council of Teachers of English, 1971.

Flower, L. Writer-based prose: A cognitive basis for problems in writing. College English, 1979, 41, 13-18.

Graves, D. H. Balance the basics: Let them write. New York: The Ford Foundation, 1978.

Graves, D. H. Writing: Teachers and children at work. Exeter, NH: Heinemann Educational Books, 1983.

Macdonald, N. H., Frase, L. T., Gingrich, P. S., & Keenan, S. A. The Writer's Workbench: Computer aids for text analysis. Educational Psychologist, 1982, 17, 172-179.

Nold, E. W. Revising. In C. H. Frederiksen & J. F. Dominic (Eds.), Writing: The nature, development, and teaching of written communication (Vol. 2). Hillsdale, NJ: Erlbaum, 1981, pp. 67-79.

Scardamalia, M., & Bereiter, C. Assimilative processes in composition planning. Educational Psychologist, 1982, 17, 165-171.

Shaughnessy, M. P. Errors and expectations. New York: Oxford University Press, 1977.

LOGO PROGRAMMING AND PROBLEM SOLVING*,**

Roy D. Pea

Center for Children and Technology
Bank Street College of Education

In the world of educational computing, programming is a major activity, occupying several million precollege students a year in this country alone. As yet very little is known about what kinds of cognitive activities computer programming requires and whether, in the classroom contexts that are representative of microcomputer use in schools today, children are capable of making substantial progress in learning to program. In the cyclical program-development process of problem understanding, program design and planning, programming code composition, debugging, and comprehension, what gains do children make on the many developmental fronts represented in the complex of mental activities required by programming? Do conceptual limitations impede their understanding of any of the central programming concepts, such as flow of control structures, variables, procedurality, and the like? We have begun to address aspects of these questions in our developmental research on children learning to do Logo programming.

I would like to make five points which will be explicated in the remainder of this paper:

1. Systematic developmental research documenting what children are learning as they learn to program is necessary, rather than existing anecdotes. Our studies focus on Logo because it is a programming environment that is exciting to many educators, it has great potential for introducing children to many of the central concepts involved in programming and problem solving, and because grand claims have been made for how it promotes learning to program, as well as metacognitive skills such as planning and strategic problem solving.

---

*Technical Report No. 12.

**The research reported in this paper was funded by the Spencer Foundation.

2. Logo is cognitively complex beyond its early steps, and quite difficult to learn without instructional guidance, even if students are intellectually engaged with that learning. While the semantics and syntax of Logo are readily learned, the pragmatics--how to arrange lines of legal programming code to achieve specific ends--is a great challenge.

3. The pedagogical fantasy (e.g., Byte, August 1982; Papert, 1980)--that Logo can serve as a stand-alone center in classrooms for learning programming and thinking skills--does not work. Teacher training will be necessary for programming skills to develop very far, and problem-solving skills may need to be taught directly rather than assumed to emerge spontaneously from learning Logo.

4. After a year's experience of programming in Logo, following the discovery-learning pedagogy advocated for Logo, two classes of 25 children (8- to 9-year-olds, 11- to 12-year-olds), each with six computers, did not display greater planning skills than a matched group who did not do Logo programming.

5. We need to develop an instructional science for programming if that is what we wish children to learn, but we also need to rethink, in ways suggested by Midian Kurland, the educational goals that programming is meant to fulfill.

The great excitement in education about children's learning to program with microcomputers is easy to understand. But it is of particular interest to me as a developmental psychologist that such excitement has had less to do with the practical value of learning how to write programs for specific applications than with the belief that, through learning to program, children will develop powerful cognitive skills such as planning abilities, problem-solving heuristics, and reflectiveness on the revisionary character of problem solving itself (Pea & Kurland, 1983). This idea--that programming will provide exercise for the highest mental faculties, and that the cognitive development thus assured for programming will generalize or transfer to other content areas in the child's life--is a great hope. Many elegant analyses offer reasons for this hope, although there is an important sense in which the arguments ring like the overzealous prescriptions for studying Latin in Victorian times.

Programming is viewed by many of its devotees as a "Wheaties of the Mind," a panacea for the ambiguities of everyday cognition. It is alleged that in the demands which programming activities make on the mind--of precision (in requiring a specific sequence of instructions for controlling the operations of the computer); of problem decomposi-

tion (into component subproblems); and of debugging (systematic efforts to eliminate discrepancies between the intended outcomes of the program and those brought about through the current version of the program)--programming renders salient the general utility of such cognitive activities in problem-solving efforts, and that such generalizations will be made spontaneously by the programmer to problem spheres above and beyond the microcomputer environment (Feurzig et al., 1981; Minsky, 1970; Papert, 1980). To place these claims in a larger context, we may note their similarity to claims about how literacy, or mathematics, or logic serve as "cognitive amplifiers," enabling the users of such technologies to transcend the limitations of their previously available tools of thought (Bruner, 1966; Cole & Griffin, 1980; Goody, 1977; Olson, 1976).

What has been done to evaluate the empirical validity of these important claims? While Papert and colleagues undertook extensive studies of children doing Logo programming in the Brookline school system, their reports of this work were principally qualitative in nature, citing and discussing some of the programs that were created by the children, the global differences in programming style that seemed to be intuitively distinguishable (Watt, 1979), and dramatic case studies of great programming progress made by children who had learning difficulties (e.g., Weir, 1981).

Though interesting, these reports do not directly address the widely touted claims for the development of thinking skills that transcend the programming context, for which case-study methods are inappropriate. We thus undertook a series of investigations in order to provide systematic data on children learning to program and the alleged cognitive outcomes of such programming, such as developments in planning skill. Methodologies for addressing these questions were developed, and summaries of some key research findings to date are presented below.

Research

I will briefly review three of our studies. Detailed technical reports will be available in the near future. The first was a study of the level of programming expertise that children had developed by year's end; the second consisted of systematic probes of the depth of understanding of programming concepts such as "recursion" in studies with individual children; and the last asked whether children doing programming developed planning skills that they then spontaneously transferred beyond programming.

In one study, we presented children with a 3-part written assessment of programming. The three parts, each taking 45 minutes, were: (1) Logo command understanding, where children were asked to fill in, with graphics or words, what would happen on the screen when specific commands were typed and entered into the computer; (2) writing Logo programs to draw shapes, with constraints as to what programming constructs (e.g., tail recursion, variable) were to be used; and (3) finding the errors or bugs in prewritten programs intended to achieve a pictorially specified goal. For command understanding, we found that, although the number of hours spent programming by the older (25 hours) and younger (29 hours) groups were not significantly different, the older group understood significantly more commands than the younger children. Boys spent more hours programming (34 vs. 22 for girls), and outscored the girls on nearly every class of programming commands. Performance on this command comprehension task was revealing: out of 100 possible points, the mean score for commands understood in terms of this measure was 34, with a huge standard deviation of 25, and only three out of the 50 children scored between 75 and 95. Roughly one-quarter of the children in each of the classes had not become very much involved in the classroom programming and did correspondingly poorly. In the case of writing different programs that would each draw a box of a certain size, we found that, while few children had difficulty writing a program consisting of a chain of direct commands (FD, RT) or a tail recursive procedure, many children could not write a version of such a program using a variable, or a version of the tail recursive program with a conditional test that would stop the drawing. In the area of debugging, many children were able to locate and eliminate "surface" errors of syntax, or missing variable values, but very few found procedural errors in which the order of lines in a program was mixed up.

The second study (Kurland & Pea, 1983) utilized a series of increasingly complex Logo programs that were designed to reveal the depth of understanding of recursion in a half dozen of the best programmers in the two Logo classrooms. The method we used--having children read through the programs line by line and make predictions as to what would happen when each line of programming code was executed--was extremely telling, and confirmed our classroom observations. Four prevailing tendencies are of central developmental significance. One was to treat the program as akin to natural language text, ambiguous in meaning and "ignorable" by the computer if the child did not understand it. A second was the fact that some children did not understand conditional test statements in these programs even though they had written programs that contained them. This is a robust finding, as other studies with these children

have shown; the children's programs often displayed production without comprehension, in that programming constructs such as variables, test statements, or even simple commands like "repeat" may have been used in one program, but not understood in another. Rote use of "chunks" from other children's programs or those of the teacher seems to be responsible for this rigidity of use. A third tendency was to violate the sequentiality of program execution, to assume that, without instructions to do so, the computer could "jump" some lines in the program to execute other lines. The fourth tendency, common to all the children, was to manifest a misguided mental model of how recursion works, one which is insurmountable without instruction since, for recursion, evidence for how control is passed in Logo (i.e., which line is to be executed next) is not discoverable in the surface structure of the language.

The third study was a longitudinal pre-post investigation of groups of children who were provided with extensive opportunity to program in the Logo language over a school year. These children were then compared to a matched group of nonprogramming students to see whether learning to program enhanced the development of planning skills. The task, administered before and after the year of programming, was a classroom chore-scheduling task that allowed children multiple opportunities to come up with the shortest plan they could construct for carrying out a series of chores. Our expectation was that better planners would take a more strategic approach to the task, revise or debug their plans more effectively, and engage in more executive and metacognitive decision making as they developed their plans (Hayes-Roth & Hayes-Roth, 1979; Pea, 1982). On a large number of measures--the efficiency of the plans, the quality of the revisions, and the types of decisions made during the planning process--we found no differences between the programming and nonprogramming groups at either age.

Why did we find no cognitive benefits on our task for those children who had been doing Logo programming for a year? Advocates of the cognitive benefits of programming might object that our treatment was not of sufficient duration for benefits to be manifested, or that benefits could be revealed in later years, but not so soon after the "treatment" provided by Logo.

However, we favor an interpretation more in keeping with two general findings in cognitive science during the past decade, and with additional observations of the children in our planning task while programming. The first finding is that transfer of problem-solving strategies between dissimilar problems, or problems of different content, is notoriously difficult to achieve even for adults (Gick &

Holyoak, 1980; Hayes & Simon, 1977; Tuma & Reif, 1980). The second finding is that, even among computer science students in college who, by their junior year, have had several thousand hours of programming study (as contrasted with about 30 hours for our student groups), great conceptual difficulties in understanding how even brief programs are working persist (Soloway et al., 1982), which one would not predict if planning and problem-solving skills had achieved such extensive development by virtue of programming experience.

Our in-class observation had to do with whether children plan prior to programming. It has been an assumption of those expecting transfer of planning skills developed within programming to domains outside programming that, in fact, planning skills are at least developed in programming. But we found very little preplanning activity. Planning a program by specifying the high-level logic that a program would be written to implement was not a distinct component of the children's program development process. Much more common was on-line programming, in which children defined their goals, and found means to achieve them as they observed the products of their programs unfolding on the screen. Rather than constructing a plan, then implementing it as a program to achieve a well-defined goal, and afterwards running the implemented plan on the computer, children would evolve a goal while writing lines of Logo programming language, run their program, see if they liked the outcome, explore a new goal if they did not like the outcome by writing a new programming code, and so on. It might be objected that, although they engaged in little top-down planning, they did work a great deal on plan revisions by continually adapting their programs, revisions being central to planning activity (Pea, 1982). If this is so, we should have seen differences between the programming classes and the control classes in planning revisions during the noncomputer planning task, but we did not. And program debugging in the classrooms would have been very common. In most cases, children preferred to rewrite a program from scratch rather than to suffer through the attention to detail required in figuring out where a program was going awry. As one child put it when asked why she was typing in commands directly rather than writing a program: "It's easier to do it the hard way." Debugging requires precision and line-by-line program comprehension; in general, both were difficult for the school-aged children to attain after a full year. They certainly were not automatic consequences of exposure to Logo.

While we believe that, on the basis of these findings, it would be premature to discard programming or Logo from the set of microcomputer uses in schools, these studies do raise serious doubts about the

sweeping claims made for the cognitive benefits of learning to program, particularly in Logo (see Byte, August 1982). We find that the entry level of Logo--getting the turtle to carry out mathematically interesting drawings through writing short programs consisting of one or two procedures--does not present conceptual problems for the school-aged child. Far from being problematic, one finds in most children just the mental engagement that advocates of Logo highlight. But the elegance and beauty of Logo that derives from its parent language, LISP, used in artificial intelligence, its procedurality which allows one to define new procedures and use them as building blocks in increasingly complex programs, its control structures that allow very brief recursive programs that can solve quite difficult problems, the use of conditional tests--all these features present deeply challenging conceptual problems on a turf our children did not opt to travel during their discovery learning. With thoughtful instruction, which will require developmental research for its design, we expect that Logo may provide a good window for the child into these important computational concepts. With accompanying instruction in thinking skills (see, for example, Chipman, Siegel & Glaser, 1983), perhaps using Logo or other programming languages as a vehicle for discussing heuristics and problem-solving methods, developments in planning skill may in fact be achieved. But we have deep doubts, based on a series of empirical studies over an 18-month period, that the Logo ideal is attainable with its discovery-learning pedagogy.

References

Bruner, J. S. On cognitive growth. In J. S. Bruner, R. R. Olver, P. M. Greenfield (EDs.), Studies in cognitive growth. New York: Wiley, 1966.

Chipman, S., Siegel, J., & Glaser, R. (Eds.), Thinking and learning skills: Current research and open questions. Hillsdale, NJ: Erlbaum, 1983. In press.

Cole, M., & Griffin, P. Cultural amplifiers reconsidered. In D. R. Olson (Ed.), The social foundations of language and thought: Essays in honor of Jerome S. Bruner. New York: Norton, 1980.

Feurzig, W., Horwitz, P., & Nickerson, R. S. Microcomputers in education (Report No. 4798). Prepared for: Department of Health, Education, and Welfare; National Institute of Education; and Ministry for the Development of Human Intelligence, Republic of Venezuela. Cambridge, MA: Bolt Beranek & Newman, October 1981.

Gick, M. L., & Holyoak, K. J. Analogical problem solving. Cognitive Psychology, 1980, 12, 306-355.

Goody, J. The domestication of the savage mind. New York: Cambridge University Press, 1977.

Hayes, J. R., & Simon, H. A. Psychological differences among problem isomorphs. In N. J. Castellan, Jr., D. B. Pisoni, & G. R. Potts (Eds.), Cognitive theory (Vol. 2). Hillsdale, NJ: Erlbaum, 1977.

Hayes-Roth, B., & Hayes-Roth, F. A cognitive model of planning. Cognitive Science, 1979, 3, 275-310.

Kurland, D. M., & Pea, R. D. Children's mental models of recursive Logo programs. Proceedings of the Fifth Annual Cognitive Science Society, Rochester, New York, 1983.

Minsky, M. Form and content in computer science. Communications of the ACM, 1970, 17, 197-215.

Olson, D. R. Culture, technology and intellect. In L. B. Resnick (Ed.), The nature of intelligence. Hillsdale, NJ: Erlbaum, 1976.

Papert, S. Mindstorms. New York: Basic Books, 1980.

Papert, S., Watt, D., diSessa, A., & Weir, S. Final report of the Brookline LOGO Project: An assessment and documentation of a children's computer laboratory. Cambridge, MA: MIT Division for Study and Research in Education, 1979.

Pea, R. D. What is planning development the development of? In D. Forbes & M. Greenberg (Eds.), New directions in child development: Children's planning strategies (Vol. 18). San Francisco, CA.: Jossey-Bass, 1982.

Pea, R. D., & Kurland, D. M. On the cognitive and educational benefits of teaching children programming: A critical look. New Ideas in Psychology, 1983, 1 (3). Elmsford, NY: Pergammon. In press.

Soloway, E., Ehrlich, K., Bonar, J., & Greenspan, J. What do novices know about programming? In B. Shneiderman & A. Badre (Eds.), Directions in human-computer interactions. Hillsdale, NJ: Ablex, 1982.

Tuma, D. T., & Reif, F. (Eds.). Problem solving and education: Issues in teaching and research. Hillsdale, NJ: Erlbaum, 1980.

Watt, D. A comparison of the problem solving styles of two students learning LOGO: A computer language for children. Proceedings of the National Educational Computing Conference, 1979, 255-260.

Weir, S. LOGO as an information prosthetic for the handicapped (Working Paper No. 9). Cambridge, MA: MIT Division for Studies and Research in Education, May 1981.

THE COMPUTER AS SANDCASTLE*

Jeanne Bamberger

Massachusetts Institute of Technology

What if we turn the fashionable computer metaphor on its head: The computer, rather than being a superbrain, teaching us with its consistent and logical "thinking," is instead a fantasy world which, like a hall of mirrors, reflects back to us images of our commonsense ways of making things and making sense. By common sense, I mean the constructions of our ordinary action knowledge by which we commonly sort things into same and different, separate and join, bring objects into existence by naming them, and otherwise make the sense we think we find. I would like to suggest that the computer world can, as a hall of mirrors, reflect this commonsense knowledge back to us in a new guise. Playing with what we make in the computer world--"generating it"--we can come to see familiar objects and actions in new ways, and we can come to a new appreciation of the intuitions with which we make the things we know best--the intuitions I will call our "sensory smarts."

How does this happen? To make something in the computer world, we must start with a description constrained by the units and syntax of a computer language--everything must be said. In the Logo computer world, description--analysis, if you wish--immediately turns into a visual or sounding artifact that may carry with it surprises. Playing with these unexpected reflections of our thought actions, we can find--sometimes for the first time--aspects of our most intimate ideas and objects, and can then learn to go beyond them.

> The aspects of things that are most important to us are hidden because of their simplicity and familiarity. (One is unable to notice something--because it is always before one's eyes.) The real foundations of enquiry do not strike a man at all. Unless that fact has at some time struck him. And this means: we fail to be struck by what, once seen, is most striking and powerful. (Wittgenstein, 1953)

---

*Technical Report No. 20.

But there lurks a tempting danger. Instead of being seen as a fantasy world of reflected images, the computer world of procedures that make clean and beautiful objects is always on the verge of seducing us--of becoming more real than the messy, unpredictable, uncontrollable world of human sensory, felt experience. How can we use the two worlds so that they inform one another instead of each being contained and defined by its own seemingly incommensurate primitives?

Let me propose a strategy. Suppose we adopt a stance as we make things in the computer world, similar to that of children, craftsmen or artists in their work of making things in the handleable, sensory world--a sandcastle, a paper doll, a design drawn on paper or one built with blocks. Then, from this process of everyday designing and making, take a mode of learning that seems particularly to characterize it. I will call it "conversational learning" (see Bamberger & Schön, 1983).

By conversation, I mean the conversations we have with materials as we build or fix or invent. As we perturb these materials, arranging and rearranging them, watching them take shape even as we shape them, we learn. The stuff talks back to us, remaking our ideas of what is possible. The backtalk leads to new actions on our material objects in a spiral of inner and outer activity. Inner intention gives way to reflection on and responsiveness to the backtalk of the materials, leading to new outer actions on objects, and thence once more to changed intention. It is a kind of research--one that is as familiar to the scientist designing a theory as to the painter or composer designing an artifact.

The painter, Ben Shahn, says:

> At one point [a painter] will mold the material according to an intention. At another he may yield intention--perhaps the whole concept--to emerging forms, to new implications within the painted surface....Thus idea rises to the surface, grows, changes as a painting grows and develops. So one must say that painting is both creative and responsive. It is an intimately communicative affair between the painter and his painting, a conversation back and forth, the painting telling the painter even as it receives its shape and form. (1957, p. 49)

Shahn's words could describe designing and making a computer procedure: "conversation back and forth, the 'programmer' telling the 'program,' the 'program' telling the maker even as it receives shape

and form." But the differences between programmer and painter are as important as the similarities.

We need to make a conversation between these two ways of world-making--the handling of materials, immediate sensory actions, and the world-making of description where we have to say what we want and then that happens. In this conversation, the computer becomes a hall of mirrors as it reflects back to us a surprising, maybe strangely unexpected, image of our everyday making and shaping. Encouraging such conversation by confronting the computer's unexpected reflected image of what we think we do with the sound or picture we expected, we may come to see strategies, kinds of things, kinds of relations that otherwise might remain hidden in the givens of each world.

We take a musical example, but it requires attention to the following:

The intention of this sounding example cannot be properly captured in the silence of words or notations on paper--it must be done live and in sound through clapping or drumming, and in interaction with the computer and its synthesizer. The very difficulty, however, helps to make the argument. If we want to use the computer to confront the mismatches between what we can do, what we think we do, and the results of these thoughts as descriptions, we must be able to hear the results of these descriptions in actual sound. It is the argument, here, that the computer is unique in making such confrontations possible. Indeed, it is in this unique capacity that the computer functions as a mirror, giving us back a sounding image of our described thoughts. And this is how: Putting our "thought actions" in the form of instructions to the computer, these descriptions/instructions are instantly realized in sound by (in this example) the synthesizer drum. If what we hear is different from what we expect--that is, different from the results of the live actions we had thought to describe--then the stage is set for learning. But if readers are given the rhythm as it is described by standard music notation, there is no opportunity left to make the experiment. Instead, you simply make an easy translation from one code to another--from music notation to computer notation. There is, then, no opportunity to say what you think and then to learn through the possible surprises as you listen back to what you said, now in sound.

As one college student put it in puzzling over what and how she was learning in this environment: "We are used to expecting answers from the computer, but here the answers have to come from us."

I will try to put the reader into the experiment by using language that does not preempt that possibility. But readers must go into

action themselves, too. To begin with, clap the rhythm of the rhyme:

Three, four, shut the door
Five, six, pick up sticks

Now, to remember the rhythm or to help another play it, make a drawing. Most people will draw two big shapes (maybe circles) and three small ones; and then again two big ones and three small ones:

O O o o o O O o o o

Now, how can we get the computer drum to play the same rhythm? The Logo turtle drum can make two kinds of drum hits--a long one called "L" and a short one called "S." Think of time as space; L and S are related to one another like this:

L: — — — — — —
S: -- -- -- -- -- --

Try clapping the Ls and the Ss:

L's sound: Tick, -- tick, -- tick, --
S's sound: Tick-tock, tick-tock, tick-tock

Looking at the drawing of the rhythm, now we translate it into turtle drum instructions using L and S:

O O o o o O O o o o
L L S S S L L S S S

And this is what we hear back:

Three, four, shut the door; five, six, pick up sticks.

L L S ⌈S S L L S⌉ S S
O O o o o O O o o o

Our description, our thought object, results in a new and unexpected sounding object:

⌈S S S L L⌉
Shut the door five, six

becomes a single, uninterrupted figure, and the repeated figures we intended have disappeared, the boundary between them gone. Using

the computer as a mirror, we find not what we intended but instead a new figure, perhaps more interesting than the one our hands spontaneously made. We can repeat it:

S S S L L S S S L L
o o o O O o o o O O

to make a kind of Latin samba.

But how can we use this surprising reflection to help us make our intended rhythm? Try to clap the intended rhythm again and compare it with the results of the description played on the turtle drum. We discover, in this conversation between what we clapped and the results of what we described, that something is missing. The two figures of our clapped rhythm are bounded, separated by a gap, a silence. The gap is missing in the turtle drum's "performance" of the rhythm. It must, then, be missing from our description.

Does the turtle know how to pause? Yes. It can be told to make an S-pause or an L-pause. We insert an S-pause between the two figures:

L L S S S [S-pause] L L S S S

It works! The two repeated figures are once more intact. By listening to the pretend turtle drum as it responds to our instructions, working with it as material, we discover, in its reflection, hidden intimacies of our own performance. What seemed in our actions to be a nonaction, a nonthing--silence--materializes as a reality that needs to be accounted for.

But is it possible to make the same rhythm using only longs and shorts--just Ls and Ss? Think about the time that goes across the boundary of two repeated figures--the time we have just filled up with an S-pause. There is, in fact, an S (the last of the three together) followed by an S-pause. The two together make a single L. We can try:

L L S S L L L S S L

That works too! It sounds the same. But this new long, even though the same in time as the others, seems a different long from those at the beginning and those that follow. It is an "ending-L" instead of a "beginning-L"--an L made of an S and an S-pause, a sound and a silence. The same measured time can be very different

depending on where it happens and how it functions--beginning as compared with ending, for instance. That's an important idea. Measured property and function make different meanings. The process is open-ended, topless. Silence becomes material; time goes on even when actions seem to stop; the same time can seem different; filling up time depends on what is happening in it.

In just this small example, surprises from making thought objects into sounding objects tell us about our own sensory smarts and give us new ideas for making things with them--new and, until then, perhaps impossible rhythms to play on a real drum. The structures and things in one world seen as the structures of the other--we can go beyond them both. This is but a bare beginning, hardly exploring the power of each. The point, then, is not to give privileged status to sensory smarts or to the procedural descriptions of a computer language such as Logo, but rather to appreciate and learn from multiple ways of knowing, doing, and saying. And, most important, to learn from these two ways of world-making, these interterrestrial conversations, where and when it is useful and productive to measure, take apart, and describe, and where to gesture and directly experiment using our sensory smarts. We develop, in fact, a kind of metaphor-in-action--two organized structures--the structures made possible by the entities and relations specifiable in the Logo world, and the structures made possible by the entities and relations accessed by our everyday knowhow. Seeing one as the other, we can breed new entities, new ideas.

When we use the computer as a pretend world of animated reflection with which to make conversations, it becomes a terrain for inquiry. The most commonsense aspects of things that otherwise remain hidden emerge, helping us to account for and to build on what we already know how to do so well but can't say. Like children, pretending can make real what is otherwise unseen and most valued.

References

Bamberger, J., & Schön, D. A. Learning as reflective conversation with materials. Art Education, April 1983.

Shahn, B. The shape of content. Cambridge, MA: Harvard University Press, 1957.

Wittgenstein, L. Philosophical investigations. Oxford, England: Oxford University Press, 1953.

LEARNING LOGO TOGETHER: THE SOCIAL CONTEXT*,**

Jan Hawkins

Center for Children and Technology
Bank Street College of Education

Collaboration in Classrooms

There are many different ways to organize learning situations. In classrooms, learning is traditionally organized around an adult who has primary responsibility for teaching information and organizing material so that novices can learn about particular subjects. Students learn and practice the presented information and skills. There are, however, ways to organize learning other than by the teacher presenting information to a group of novices. For example, in many learning situations, novices are apprenticed to experts who communicate the information needed for a certain skill, and who support practice in that skill. This occurs often with job-related training (e.g., architects learn the tricks of their trade from experts), and crossculturally in what are sometimes called informal learning situations.

There are also a variety of learning arrangements in the group-learning environment of the classroom, based on alternative theoretical models of how children learn. I will discuss two of these models--one by way of example, and the second as the major topic of this paper. First, a constructivist model of development was the foundation for the notion of discovery learning. When adapted to classrooms, discovery-learning models characterized children, in a sense, as their own teachers. With the support and guidance of adults and a well-arranged environment, children explore and learn in accordance with their own pace and needs. In this ideology, the child is seen as a self-motivated individual who learns through discovery when allowed to select his own material world for engagement. This discovery-

---

*Technical Report No. 13.

**The research reported in this paper was funded by the Spencer Foundation.

learning model is the basis for the pedagogy associated with Logo--children invent their own goals to accomplish in the programming language of the microworld. They discover the necessary tools, as needed, for self-directed purposes.

There is another, less well-defined learning organization for classrooms, based on different ways of thinking about the social situation and the social resources for learning; that is, that important learning occurs when children do collaborative work, and when peers teach each other. The notion of the importance of teaching someone else in order to learn has been around for a long time. In 1632, Comenius wrote that the process of teaching gives great insight into the subject taught: "If a student wishes to make progress, he should arrange to give lessons in the subject he is studying, even if he has to hire his pupils." In the late 18th century, the increased interest in public education made the student-teaching-student model economically attractive because there were too few teachers. Lancaster turned this to his advantage, and demonstrated good results from learning through peer-teaching. This was popular in the United States at the beginning of the 19th century and continued for about 20 years.

When one child teaches another, it is commonly assumed that the learning goes both ways; the experience is at least as beneficial to the teacher as it is to the student. Peer-tutoring programs have been organized in many schools for a number of years, and there has been federal interest during the last few decades in exploring its possibilities for different subject areas. These are commonly organized as formal programs outside of classrooms, and there is research evidence that peer tutoring is a viable model for learning.

However, the organization and benefits of the collaborative work/learning situation, or less formal peer teaching, that occurs in classrooms are less well defined. Collaborative work involves individuals in jointly producing a project or goal. While many teachers value this kind of interaction among children, the benefit is often expressed in terms of learning interaction skills, rather than learning information or subject matter.

The notion that collaboration on work can be an effective learning situation is also based on developmental theory. Perhaps the best theoretical expression of the idea that knowledge is acquired through the internalization of social forms is that of Vygotsky. In his view, cognition originates in social interaction and can be described as internalized dialogue. The social context supports learning and cognitive change: skills first available and "scaffolded" jointly with another person become available at an intrapersonal level. This

framework extends to a variety of learning situations--from that of the formal classroom to the more informal collaborative exchanges.

What Has Been Claimed for Learning from Socially-Based Classroom Organizations?

Peer teaching and collaboration situations are valued by many teachers here in the United States and elsewhere (e.g., Japan). A variety of claims have been made about the importance of these types of learning contexts, but very little research has been conducted to evaluate these claims. With respect to cognitive issues, there are three kinds of claims:

1. Cognitive skills are enhanced. Children learn better or differently. This claim is frequently grounded in Vygotsky's ideas; it also appears in recent Piagetian research. Conflict in viewpoints can invoke cognitive change, particularly in children transitional between stages. Finally, there is the functional notion that requiring children to verbally formulate information improves their understanding of the material.

2. Metacognition is enhanced. In a collaborative or peer teaching context, children learn about learning.

3. Children learn something about the nature of information--that it can be represented in different ways and organized flexibly, depending upon one's purpose.

With respect to social issues, there are several claims made about the value of this learning context:

1. It enhances social interaction skills by facilitating working together and thus communicating the value of learning skills for joint efforts.

2. It contributes to children's positive views of themselves and their own competence.

3. It teaches children how to make use of resources available to them for help. The classroom is organized so that children are aware of social resources and learn the skills of selecting and obtaining help.

4. Finally, benefits have been claimed for the school community (as in the 19th century). Encouraging this sort of social organization

of work in classrooms increases the amount of individualized instruction that occurs.

But, as noted above, there has been relatively little research done to investigate the claims about the importance and impact of collaborative work for learning and learning environments. At the most basic level, there is very little known about the occurrence of such social forms in classroom contexts.

Where Do Microcomputers Come In?

As discussed by Roy Pea, we have been working closely with two classrooms in order to understand how children learn and work with the Logo programming language, and how the technology is incorporated into the life of the classroom. I will discuss the second of these topics: the relation of microcomputers to the classroom as a social context. Microcomputers also appeared to offer a particularly good opportunity to look closely at the nature of collaborative work among children.

In an earlier survey study conducted by Karen Sheingold and associates, it was found that many teachers remarked on the social changes occurring in their classrooms when microcomputers were introduced: there was more peer collaborative work going on and more use of children as resources for each other. This was something of a surprise, given the common picture of the computer "hacker" as a social isolate. But the fact that many teachers remarked on this phenomenon interested us, and we decided to pursue a series of studies concerning the social features of microcomputer work for children.

In addition, there are rational reasons to expect that this technology would elicit information exchange among children: the public nature of the work--its availability to any passing viewer on the vertical CRT screen; the explicit nature of the problem steps; the limitation of resources for solving problems in programming; the "videogame" culture which normally involves groups of children in play. Everyone, including the teachers, was a novice with the machine, and the computers were a novelty in the classroom; there was no shared understanding of expectations for work or curriculum in this area.

What Did We Want to Find Out?

Based on assumptions about the occurrence, nature, and benefits of peer collaboration and teaching, we asked a variety of questions in our studies about the organization of learning in the classroom, children's working-together skills, and the impact of microcomputers.

First, we wanted to know about the occurrence, number, and occasions of peer work-related interaction in these classrooms. The teachers valued children's working together, encouraged them to do so, and designated particular times of the day as work periods when children could do their own or group work. We documented the number of peer interactions that occurred in work settings. We conducted two observational studies in the classrooms.

Second, we were surprised to discover that virtually no literature is available which examines children's understanding of the classroom as a work context. This is the place where children spend a large portion of their waking hours, and the one which powerfully instructs them about the nature of work in our culture. We felt that it was important to understand children's perceptions, understanding, and preferences about the social organization of their work context. Consequently, we interviewed children in each classroom about this at the beginning and the end of the school year.

Third, we wanted to know more about children's skills in doing collaborative work--the process of producing joint solutions to problems in both Logo programming and other tasks. We videotaped same-sex pairs of children several times over the course of the school year. The tapes were analyzed to determine the problem-solving strategies, and the nature of the interactions that produced the results. In one study, we also wanted to know what individuals learned from the joint effort.

Finally, the social history of the technology in the classroom was documented by ethnographic observations over the course of the school year. How do the learning context and the curriculum evolve? These observations were supplemented by regular interviews with teachers and students.

I would like to comment principally on the first two issues: the occurrence of collaborative work in the classroom, and children's views of this work context.

First, time-sampled observations were done during work periods when children were engaged with both computer programming tasks and noncomputer tasks (such as math, language arts, map-making). The occurrence of task-related and nontask-related turns of talk were recorded, as well as the events of peer teaching or collaboration (either verbal or action). One study was conducted over a 6-week introductory period at the end of one school year (before and after the computers were introduced). The study was replicated, with some revisions, over the course of the next school year. The find-

ings were consistent for the two studies: children in both classrooms tended to talk to each other much more about their work when they were doing programming tasks than when they were doing noncomputer tasks. At both these times, children were free to work with someone else or by themselves. In addition, children engaged in more collaborative activity with computer than with noncomputer tasks. Peer teaching occurred very infrequently in any context. In contrast, children did talk to each other when they were working on other classroom tasks (e.g., math); however, what they talked about was often not related to what they were doing (e.g., vacation, what they would have for snack).

According to this observational work, peer collaboration and teaching did not occur particularly frequently in these classrooms for traditional classroom work, despite the fact that contexts were organized to encourage such exchange. It should be noted that there are certain projects that require collaborative effort, such as putting on a play or doing a group mural. However, we were most interested in comparisons between tasks (like the computer ones) where children could choose to do the work either individually or collaboratively.

The computers appeared to provide a context where collaborative work was supported. We suspect that this was due to a variety of things (noted above). An explanation based simply on the novelty of the technology can be eliminated because the differences between computer and noncomputer tasks persisted over the course of the school year. In addition, it should be noted that the amount of task-related exchange between peers did not change over the course of the school year for noncomputer tasks; there was no general change in this aspect of the learning environment.

We are currently engaged in some studies to determine how different aspects of the computer context may support collaborative efforts. Other types of observations that we have conducted indicate that there are at least three types of joint peer engagement with computers: (1) sustained collaboration on a joint project; (2) seeking help or advice for a problem; and (3) "pit-stopping," where children traveling around the classroom drop in at a computer "for remarks." In all these forms, children provide support for each other in accomplishing their work.

Second, we wanted to know about children's perceptions of this work context in general, with particular attention to the presence of the microcomputers. In the interviews, we asked children a series of questions about four overall topics: (1) their perceptions of the social organization of the learning environment; (2) their preferences

for work arrangements; (3) their understanding of what made people good collaborators or teachers; and (4) their understanding of what is involved in the process of collaborative work. When the interview data were analyzed qualitatively in order to understand the dimensions of children's understanding, some overall themes emerged. While there was some variation among age groups, several general points may be made from this analysis:

- Children clearly discriminated between occasions for working together and alone in their classrooms. For the younger children, this varied by subject area: research and math were the most frequently cited occasions for working together. Among the older children, the occasions for collaborative versus solitary work varied according to both the nature of the work and the subject area. For example, some children reported that joint work occurred when something was particularly difficult (they'd ask questions of a peer), or when the work was mechanical.

- Most children had clear preferences for working alone or with someone else. While some always wanted to work alone and a few always preferred a partner, most children offered criteria for when one or the other situation was preferred. Preferences for solitary work were dominated by a concern for getting something done with speed and efficiency, for not wanting to take time to explain to or negotiate with someone else. Some also felt that working with someone else had an aura of cheating about it--you shouldn't let the teacher know you're doing it. In addition, some children felt possessive about their work or ideas and did not want to share them with another person.

  Preferences for working with someone else were dominated by social concerns (e.g., more fun, you can be with a friend), or by anticipating occasions when you know you're going to need help with something (the other person can provide help and/or ideas). Thus, children tended to see collaborative work as appropriate on two kinds of occasions: (1) when the primary activity was social (fooling around, having fun); or (2) when help was needed with a particular aspect of something being done by a single person. In this latter case, children asked for information rather than for collaborative assistance. It is noteworthy that children did not discuss collabora-

tion as the joint effort to learn or produce something as a sustained interaction.

- Children tended to dislike being assigned to collaborative work by the teacher, both for reasons of mismatch between people (one of the main reasons for working together is that it's more fun with a friend), and because of inferred teacher intent (some suspected that, in making assignments, the teacher was trying to encourage social relationships between children, having nothing to do with getting the work done efficiently).

- Most children had relatively sophisticated notions about the characteristics of good collaborators or helpers, and what was entailed in the process of good collaborative work or helping efforts. For example, the ability to explain was the most frequently cited asset of a good helper, someone you go to for assistance ("You can know a lot and be a friend and still not be able to help"; "You have to be able to say it well"). In addition, age made a difference: teachers or adults were the source of information about "big" things (such as explaining how something works), whereas peers could help with more limited things (such as how to spell a word). Good explanation involved a combination of ideas about telling and demonstrating: it is done in a way that allows the learner to understand; that is, the "teacher" takes into account what the learner already knows.

- With respect to the presence of computers, most children reported that this was a context, more than any other, where children worked together and would go to each other for help. Indeed, many of the programs and projects produced were joint efforts to some degree. This is probably due to several things: First, it was a new subject for everybody, and different children had different "pockets" of knowledge about features of the programming language; they traded information and programs fairly freely. Second, there were limited teaching resources--one teacher for six machines--so if you didn't want to wait in line with your problem, you found help elsewhere. Third, while it is unusual to have six machines in a classroom, it is still a limited resource in a class of 25--children could share their computer time with each other, and therefore work more. Fourth, as discussed above, features of computer tasks seem to support

collaborative engagement. Finally, the computers had not yet been adopted as a legitimate classroom subject; thus, their status with respect to the efficient work/fun distinction had not been established. At the beginning, the kids thought of them as game devices.

There is something of a contradiction in this group of findings (the observations and interviews) which reflects, I think, some implicit contradiction in the communicated purpose and value of collaborative work. Overall, children have a fairly sophisticated understanding of the social organization of their learning context, what collaborative and helping efforts are all about. However, our observational data indicate that they do not frequently engage in such activity. In addition, the interviews indicate that many children are not clear about the value of collaboration in getting something done--working together may make the effort more fun, but probably hinders getting the work done efficiently. Thus, children know what this work form is, but don't do a lot of it because there appear to be other priorities in the learning context.

However, the observational and interview studies demonstrate that the computers are a context in which children engage in and appear to value the effectiveness of collaborative work in getting something done. The technology, therefore, offers the possibility of a classroom learning context where efficient collaborative activity might occur with some frequency. As Cindy Char will discuss further, the technology only offers a framework. The learning opportunities available to children hinge on the ways that teachers choose to use them.

I want to end on a speculative note. The frequency of collaborative activity with computers may also, in part, be related to the indeterminate status that the computer activity had in the classrooms: Is it legitimate, accountable work for which individuals are responsible, or is it supplementary, fun activity? The teachers were unclear about the computers and programming as a part of the curriculum and modified their views as the year progressed. In general, however, throughout the year the children were not held responsible for learning and knowing particular aspects of the language; features were learned as they became necessary for projects. The computer "subject" did not have the same status as the other curricular areas in terms of required knowledge or legitimacy, and the kids tended to view it as a supplementary activity. The observed jointness of this activity may then be partly related to an historical accident: computers/programming are new arrivals in classrooms. The role of the computers and accompanying software in terms of legitimate work has

to be determined. As the computers acquire the same status as other subject areas, the individual work responsibility associated with such knowing may have an impact on the organization of the learning context for the technology.

RESEARCH AND DESIGN ISSUES CONCERNING THE DEVELOPMENT OF EDUCATIONAL SOFTWARE FOR CHILDREN*,**

Cynthia A. Char

Center for Children and Technology
Bank Street College of Education

I would like to discuss a number of different research and design issues that need to be considered when creating educational software for classroom use by children. My comments stem from a fieldtest evaluation that was conducted on three types of innovative software created at Bank Street College, which were produced as part of a multimedia curriculum package on science and mathematics for fourth through sixth graders.

At the Project in Science and Mathematics Education, we are producing a television series, microcomputer software, print materials and, eventually, videodiscs that provide an integrated approach to science and math instruction. At the core of the Project is a 26-episode dramatic series for television, called "The Voyage of the Mimi." The series follows the adventures of two young scientists and their teenaged crew who are studying whales while aboard a research vessel. Like other media components of the Project, the series is designed to provide students with an appealing and compelling view of what it is like to do science and to be scientists, and how mathematics and technology can be used in scientific inquiry.

In order to engage children in some of the ways computers are used in the world by adults, three different types of software have been developed to accompany the television series. One software piece, Probe, displays the computer's usefulness in data collection and representation and, at present, is a software package for measuring and graphing temperature data. Eventually, the software will be able to gather data on light and sound as well. The second piece of soft-

---

*Technical Report No. 14.

**The research reported in this paper was funded by the Department of Education, Contract # DE-300-81-0375.

ware, Rescue Mission, is a navigation game that illustrates the computer's use for simulation. It motivates students to apply such skills as map reading and mathematical knowledge of scale, degrees, and angles to the real-world problem of ocean navigation. The third type of software consists of two games, Whale Search and Treasure Hunt, which introduce the notion of programming to children. To play these games, children must learn the basic commands of the Logo language to move their "ship" about the computer screen, either to reach a trapped, netted whale or some hidden treasure. Given the limited computer resources that exist in most schools, all four software pieces were designed for use by more than one student at any given time; for example, the navigation simulation game can accommodate up to 12 student players at a time.

Evaluating software designed for school use, as opposed to home use, had certain implications for the way we went about studying the effectiveness of the software. First, we were interested in seeing how the materials were used in natural classroom groupings and settings, rather than by individual or small groups of children outside the classroom. Second, we placed a special focus on teachers and their role in the classroom, rather than an emphasis only on children and their reactions to the software. We were particularly interested in seeing how teachers viewed our materials, made decisions about how to use them, and evaluated the software experience.

In order to take an intensive look at how computers are used in classrooms, and how software use is affected by teachers' views and roles, we conducted a fieldtest using a case-study approach. Thirteen classrooms, drawn from seven schools in the New York metropolitan area, participated in the fieldtest. All the classrooms were in the Project's targeted age group of fourth through sixth grade, and were diverse with respect to ethnic, class, and urban/suburban variables. The fieldtest was unstructured so as to give teachers flexibility in using the materials, and in their selection and organization of classroom lessons.

To evaluate the effectiveness of our software, three different aspects of the materials needed to be examined: comprehensibility, appeal, and usability. Furthermore, we wished to address each of these aspects from the perspectives of teachers, students, and our own staff of researchers. For example, in order to obtain a collective picture of student comprehension of a piece of software, teachers were asked what aspects seemed unclear to students; students were asked what questions they had about the materials; and students were administered a written test focusing on specific software tasks which we, as researchers, felt might be problematical for children.

Seven different types of measures were used in the fieldtest. These included classroom observations, student and teacher interviews, written forms where teachers described and evaluated the Project activities used each day, log-in books where students recorded their names and time spent at the computer, evaluation forms where students indicated their reactions regarding the software's appeal and comprehensibility, and teacher background information forms where teachers described their past teaching experiences and the science and mathematics instruction in their current classrooms. There were three different versions of each measure, one version for each type of software.

The most striking finding of the fieldtest was the considerable range of use of the software in different classrooms. Some of these differences were quantitative, such as the proportion of students in a class having access to the computer, and the amount of time each student used the software. Other differences were qualitative, such as the degree and type of teacher involvement in the software experience, and the ways teachers organized their classrooms to allow some children to work on the computer while others were engaged in noncomputer activities. For example, for the temperature gathering and graphing software, some teachers assumed the role of demonstrator, thereby engaging the whole class but limiting the amount of hands-on time students had with the computer. Other teachers chose a less central role and acted as resource persons for students working independently on temperature experiments at the computer. Still others acted as "software managers" and mainly ensured that the students had the necessary materials, that the software was working properly, and that students were taking turns in a relatively fair manner.

The amount and the way software was used in classrooms appeared to be greatly influenced by two factors. The first factor involved computer and teacher resources, and the ratio between students and computers and between students and teachers. In classes with more than one computer or with regular access to a computer, proportionately more students were able to use the software for longer periods of time, and were able to take greater advantage of the learning experiences afforded by the software. Similarly, in classrooms where there were only 15 to 20 students, as opposed to 35, teachers were able to take a more active role in the software experience--not as demonstrators, but as active participants who could facilitate and monitor children's progress with the software.

The second factor involved teachers' prior training in and perceptions of science, mathematics, or computers. For example, some of the

teachers who used the temperature graphing software were those who had had previous training in science, and who taught science as a sequential and coherent curriculum, emphasizing experimentation. These teachers tended to find the software very useful, recognized its many applications, and allowed children to work as teams of scientists and rotate through the computer station to gather data for their temperature experiments. In contrast, other teachers participating in the fieldtest had little formal training in science. They taught in schools with little emphasis on science instruction, and usually taught science as a series of lessons on various topics such as plants or animals. These teachers also liked the temperature software, but had difficulty seeing it as a flexible data-gathering and graphing tool; they were less able to generate activities and experimental contexts for the software beyond those provided in the manual. They also seemed to be less able to follow through with student questions about heat and temperature, or to notice when children were having conceptual problems with the software.

Interestingly, computer experience per se on the part of teachers did not guarantee software "success." One of the teachers who used the navigation simulation was a math teacher with prior experience in teaching children computer programming, but didn't know quite what to make of the software. He chose not to become actively involved in monitoring or facilitating students' progress with the simulation, having viewed it as an experiential learning activity with a self-sufficient and self-contained context. He also felt that the software was designed to teach navigation, rather than map-reading skills and math concepts such as angles, degrees, measurement, and triangulation. Stated another way, he felt that navigation was the curriculum content, rather than the vehicle through which to teach various math skills. Thus, our simulation and temperature gathering software may fall in a curious spot somewhere between structured CAI and computer programming, and may not automatically rest in a conceptual "niche" of computer experiences commonly found in schools.

What are the software design implications of these findings? First, to address the scarce computer resources and less-than-optimal student-teacher ratios found in most schools, it is important to begin creating software which can be used by more than one child at a time, and which allows students to work in a collaborative fashion independent of the teacher. To ensure effective use of the software, its comprehensibility to students needs to be addressed at a variety of levels --from their understanding of the text and graphics presented on each screen, to their understanding of the software's general objective, the purpose of each menu item, and how to "get around" the program. Comprehension can be facilitated by paying careful atten-

tion to the way specific screens are designed, programs are structured, and student/teacher manuals are written, with obvious advantages inherent in software and manuals which incorporate various skill and conceptual levels.

If software is designed to meet this range of student needs, teachers will be better able to work with students at a higher level of learning strategies and outcomes, and to help them make connections to other classroom activities, rather than having to explain specific text on the screen or what key to press. Teachers will also be freer to work with students engaged in noncomputer activities. Furthermore, such software will allow students to help each other and to solve problems by themselves, fostering collaborative work, as well as possibly speeding up students' time at the keyboard, thus providing a greater number of student turns at the computer.

To address the needs of teachers with limited training in or narrower views of science, mathematics, and computers, one should provide teachers with a conceptual framework for the software. This framework should describe the software's approach, educational objectives, and specific math and science skills and concepts, as well as outline connections to other activities and materials in science, math, and other subject areas. This could be done in a teacher manual or guide, or as part of a videotape or written presentation for teacher training sessions. Background material on various math and science content areas or references should also be available in the teacher's guide.

These design implications are currently being incorporated into the development of the Project software, which will undergo revision before being subjected to yet another round of formative testing in schools. Also under way are plans for the Project's teacher training component, which will address the importance of teachers and their role in classrooms when trying to implement innovative uses of educational software. With educational software and the presence of computers in schools still in their infancy, it is difficult for software developers to utilize the computer's graphic and interactive capacities and take students along new and exciting educational avenues, while still being sensitive and accountable to the needs and interests of teachers. The Project in Science and Mathematics Education, with its staff of producers, curriculum specialists, and researchers, models an attempt to bridge the worlds of the software designer, the educational visionary, and the classroom teacher. It is our view that we, along with others, should continue this pursuit to create innovative educational software that can be used in classrooms today, while exploring new directions for software to be used in the classrooms of the future.

DISCUSSION

James A. Levin

Laboratory of Comparative Human Cognition
University of California, San Diego

The opening remarks by Karen Sheingold are very appropriate. When talking about a new medium of instruction, we are dealing with a changing and change-making entity. Both Midian Kurland and Janet Kane pointed to examples of the tendency for people to think about and use this new medium in much the same way that they used the previous one (Midian focused on teachers, Janet on students).

People naturally treat a new technological development like the older one it replaces since that is the easiest thing to do. But no new development has exactly the properties of the old; thus, we get inappropriate transfer and unexpected side effects. Let us take as an example the development of the automobile. Initially, this development was called "horseless carriage," defined in terms of the previous technology. Some early automobiles even had buggy whip holders. Very few early developers foresaw drivein movies or the impact on adolescent sexuality.

Much of the early (and current) educational software still contain the vestiges of books and workbooks. The most extreme examples are the drill-and-practice programs that draw the outline of a book on the screen. Similarly, we are beginning to see unexpected side effects of computer use. Midian raised the issue of how spelling checkers should be used in the classroom, pointing to the ways in which they might modify our whole notion of the importance of spelling. Yet, we have found that, at least in the short term, spelling checkers actually increase a writer's concern for spelling, since they reduce the problem of checking a long text for errors to a more manageable one of checking a few words.

There are many metaphors for thinking about computers. The one mentioned most often in this symposium is that of "computer as intellectual tool." This metaphor certainly sheds new light on the endless discussions of "which computer language is better." Which tool is better, a screwdriver or a hammer? Certainly, that question depends on whether your goal is to fasten with nails or screws.

The "tool" metaphor is also useful in understanding the research on Logo use reported by Roy Pea and Jan Hawkins, and the work on writing reported by Janet Kane. Tools alone don't make a craftsman. A child provided with the finest workshop in the world may craft some interesting projects, but will most likely get bored and leave the tools lying about to rust and gather dust.

What is needed for effective tool use? Craftsmen are made in apprenticeships, not just workshops. In order to have learners acquire expertise in (and thus control over) these new instructional, artistic, and communications media, we must provide social systems--learning environments within which novices can progress to expertise in some systematic way, and be given technological support along the way.

For example, several studies--including those reported by Jan Hawkins and Cindy Char--have pointed to the importance of peer interaction for effective computer use. The rapid rate of technological change creates the anomaly that expertise is widely distributed in the classroom, instead of the more usual linear ordering of teacher -> smart students -> middle students -> dumb students. We find that many students know things that their teachers don't, and that the linear ordering of students shifts as different students acquire different kinds of expertise. New kinds of learning environments are needed to cope with this new educational reality.

A provocative point raised at the end of Jan Hawkins' paper raises the "specter" of the Hawthorne Effect. Will the rich peer interaction found with current computer use just be a transitory effect, disappearing when computers become so institutionalized that their use is seen by both teachers and students as classroom "work"?

In summing up, I want to make two points. First, the Hawthorne Effect is one of the largest effects in the social sciences; therefore, we should go with it as far as it takes us in the direction of effective educational uses of computers. Second, the rate of underlying technological change is much faster than the institutional change to accommodate to it. What may distinguish the computer from previous new educational technologies (which have a pretty dismal record of failing to make substantive change) is that we may have at least several decades of a continual Hawthorne Effect. This "meta-Hawthorne" effect implies that we have to take seriously the deeper implications for education of this chameleon, even as it changes colors under our very eyes.

DISCUSSION*

Joseph Glick

Graduate School
City University of New York

One of the dreams of any experimental science is to achieve that state of grace often called "prediction and control." The way that people attempt to do this is clearly to specify a set of conditions and a set of condition-relevant responses, and to demonstrate some clear contingencies linking the two. This desire for orderliness seems to be shared by anyone opting for the experimental approach whether their theoretical predilections are structural, functional, or somewhere in between. What varies among theorists are the specifications as to what constitutes a relevant description of the stimulus, stimulus-relevant response, and contingency. The issue that divides us is the "depth" of representation of these elements (surface structure vs. deep structure).

We have had a long history of pursuing this dream, and time and time again have fallen afoul of the way that human subjects seem to go about their affairs. It seems that the problem stems from the fact that humans (at the least) recode the environment, and offer responses that have some meaning to the respondent that may not be shared by the experimental paradigm. Clifford Geertz has eloquently described this state of affairs and has pointed out that we need what Ryle has called "thick description," or an interpretive (as opposed to an experimental) approach in order to adequately understand human phenomena. It seems that people have intentions and significances, and this louses up predictability. I am vastly reassured that we seem to be running into this problem again in computerland.

It seems to me that all the papers in this symposium attest to the fundamental indeterminacy relationships that intervene when we try to talk of computers and their organizational demands as providing stimulus grist for a cognitive mill. What the papers seem to attest to

---

*Revised version of remarks made at AERA, Montreal, Canada, April 1983.

is that the claims of structuralist true believers have to bow to the fundamental problem posed by human meaning-making abilities.

As the issue seems to have been put at the outset of this project, is that the computer poses fundamentally new opportunities for and demands on the child. At the social level, the computer screen is "public," capable of being commonly oriented to by more than one child. At the cognitive level, the programming of a computer necessitates the decomposition of ordinary experiences into subunits and constituent parts, and the skill to recompose the parts into well-structured, well-sequenced wholes. Additionally, since computer work is "lightweight" (not requiring physical enactment before a result is known) and "flexible" (allowing for easy rearrangement and redoing of steps), one might expect increasing flexibility of thought to ensue.

The logic of these expectations is that there may be some direct mapping between the design characteristics of the computer environment and the design characteristics of the cognitive structures and patterns of social interaction that kids in computerland might show. Whether one opts for an acquisition model stressing "structure extraction," "pattern recognition," or a model that is more interactive and stresses environmentally constrained interaction and construction based on that, there is, at the least, the general expectation of a partial isomorphism of computer and child thought and action.

In the aggregate, the results reported here suggest that these expectations have not, at least as yet, been borne out. In the main, it seems as though the computer and its created environment have been assimilated to the ways that kids usually go about doing things. Thus, the word processor is used as one would a fancy typewriter, with normal patterns of writing and composition being maintained despite the text-rearrangement features uniquely afforded by the word processor. Similarly, the opportunity for jointly attending to a common, publicly shared intellectual problem is not taken up; rather, it is social interaction as usual (with maybe a little more noise). Finally, it seems that the cognitive gains expectable from programming's organizational opportunities and demands do not show up or carry over to planning abilities nor, to go even further, are the organizational possibilities within programming themselves exploited.

It seems that the new and extraordinary environment provided by the computer is assimilated to ordinary ways of doing things. Why?

There is one way of seeing these results as not particularly unexpected. The Piagetian concept of assimilation, in one of its "read-

ings," can be applied here. In one sense, assimilation is a very conservative concept, indicating that the organism will initially treat anything new as an instance of the old. In other words, we might expect that an ordinary structure of actions would be applied to new domains of activity.

To go further with the Piagetian analogy, we would expect accommodation to occur when the ordinary structure of assimilatory actions does not, and cannot, work.

Any Piagetian reading the paragraphs above would recognize that the description of assimilation and accommodation only partially relates to Piaget's meanings. Within Piaget's theory, assimilation has often been accorded an active, "discovering" role in the psychic economy. As used in concert with Baldwin's notion of "circular reaction," assimilation is seen as a principle whereby a succession of known actions is applied to a common object, thus creating the possibility of the organization of schemes (actions) with respect to one another, modified and constrained by the nature of the environment (the accommodative aspect). Seen in this way, assimilation is an exceedingly important principle, accounting for knowledge which transcends the surface appearance of things and which is highly structured at a deep level. (Here, the added principle of reflective abstraction also allows for the organization of actions to be reflected upon and the pattern of these actions to be extracted as a structural principle.)

The results presented in this symposium serve elegantly to point out the conceptual tension that exists between the two notions of assimilation: the conservative and the progressive. The project was generated within the conceptual framework that stressed the progressive, self-discovery notion of assimilation. The results suggest the conservative interpretation of the concept.

I have no doubts that both notions apply...somewhere. The problem we face is the need to develop concepts that can help us to gain the needed analytical power to understand when they will apply, and where.

I can only make some suggestions here. First, it seems that assimilation combines two elements--the element of patterned action or scheme, and the element of intention (the use to which the scheme is put). I believe that most treatments of assimilation have given major weight to the action structure. This must be corrected by giving sufficient weight to the intentional aspect. While actions can be "structured" by the environment (or at least severely constrained by it), intentions can only be somewhat and partially impacted on by the

environment (by setting up the environment to invite certain kinds of actions), but it is not completely determined by it (people who can turn down invitations and even throw their own parties).

If we recognize the intentional aspect and its partial uncontrollability (and hence its indeterminacy), then it is incumbent on us (as researchers of behavior) to include measurements in our research that can allow us to understand more fully what people are "up to" when they are "up to" it. In other words, we need anchor point measurements which might allow us to understand the goals that a subject is pursuing. The point here is that an understanding of goals is important before any structural predictions can be made. This point has been discussed in detail in a paper delivered elsewhere (Glick, 1983).

A second feature of the reorientation toward goals is a reconceptualization of the relationships between computer environments and thinking.

The original conceptualization likened the computer to a highly structured stimulus environment whose structure would eventually occasion changes in the cognitive and social structure of kids. Ultimately some form of isomorphism between computer and kid procedural rules was expected.

A revised view of this is presented in the papers by Kurland and Bamberger. In these papers, the computer is not regarded as a structure, but rather as a tool. The focus is on the new things about the world that can be revealed (perhaps uniquely) by the computer as tool. Thus, rather than being the topic of cognition, the computer is seen as its tool. In this view, the computer becomes a source of new kinds of information, revealing nonobvious features of our experienced world, much in the way that a microscope might open up hitherto unimagined worlds. Bamberger's paper, in particular, exposes this sort of phenomenon.

In opposing the structure view and the tool view of the computer, we can gain some purchase on what may have been going on in the project reported on today. It seems that the computer-as-structure view conceives of the computer and its software as an environment for learning that teaches about itself. This view regards the computer environment as sufficiently rich to be itself an object of inquiry (and hence a rich field for discovering progressive assimilation). There is very little in the data presented here to support this view. If the computer itself is not a learning topic for the child, and we have oriented our measures to text for the learning of computer as topic, all that we can see is conservative assimilation. For, in the failure of

the child to pick up the new structure, we also see the assimilation of the new structure to old ways of doing things.

However, if we look at the computer as a tool and not as a lesson about itself, we might be able to understand its impact in a different way. Under the aegis of this thrust, we would take seriously as a research topic the documentation of the impact of the information encountered through the computer. As Bamberger has done, we would want to know about the novel form of representation that is necessitated by representation for a computer. This need not be seen as any radical restructuring of thought, but rather as a re-structuring of information.

Additionally, we might, by reconceptualizing the problem and recognizing that it is the intention to know about that governs what will be learned, begin to think of ways that the structure of the computer can be made to be a topic for learning. If we drop the pretense that computerness will automatically be "picked up," then we can seriously consider how it can be taught.

Some of this has been already touched on and signaled by the papers presented in this symposium. But much more needs to be done to explore more deeply the relationships between computers, kids, and cognition.

References

Glick, J. Heinz Werner's significance for contemporary cognitive developmental psychology. Paper presented at meetings of the Society for Research in Child Development, Detroit, Michigan, April 1983.